TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 - Elektrotechnika a informatika
Studijní obor: 2612R011- Elektronické informační a řídicí systémy

Vývoj nových modulů Domino

Development of new Domino modules

BAKALÁŘSKÁ PRÁCE

Autor: **Milan Syrový**
Vedoucí práce: Doc. Ing. Zdeněk Plíva, Ph.D.
Konzultant: Ing. Leoš Petržílka

Počet stran textu ……. 40
Počet obrázků ……. 15

V Liberci 28. 4. 2009

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Chtěl bych tímto poděkovat vedoucímu bakalářské práce
Doc. Ing. Zdeňkovi Plívovi, Ph.D.
za odborné vedení a cenné rady během konzultací.

Abstrakt

V této bakalářské práci byl řešen problém z oblasti vývoje nových modulů pro stavebnici Dominoputer, která je velmi často využívána k výuce předmětů zaměřených na elektrotechniku. Je zde popsána konstrukce dvou nových modulů a navrhnuto jejich použití.

Prvním zpracovávaným modulem je koncový modul pračky. Je to jednoduchý simulátor skutečné pračky. Na panelu jsou Led-diody, které signalizují, jestli je pračka zapnutá, stav dvířek (otevřená nebo zavřená), hladinu vody, a teplotu vody. Točení bubnu je indikováno osmi Led-diodami uspořádanými v kruhu. Tyto diody jsou řízeny programem v čipu 1016EA, který obsluhuje činnost diod. Je to v podstatě 8 úrovní řízení. Jsou to tři úrovně rychlosti, každá ve směru hodinových ručiček a v protisměru, dvě úrovně pak slouží pro zastavení bubnu.

Pro psaní řídicího programu pro tento modul a pro jeho nahrání do čipu byl použit program ispExpert firmy Lattice. Tento program používá programovací jazyk VHDL.

Druhou zpracovávanou úlohou je modul maticového bodového Led-displeje. Je zde zpracován kompletní podrobný popis konstrukce a řízení komunikace tohoto displeje řídicím obvodem. Modul byl součástí výherního automatu a jedním ze zdrojů informací se tak stala servisní příručka tohoto zařízení (připojení obvodu displeje). Závěry v této práci jsou výsledkem rozboru katalogový listů součástek a provedené simulace obvodu v simulačních programech Multisim a Proteus-Isis.

Klíčová slova: Dominoputer
Programovatelný logický obvod
VHDL
Maticový LED-displej
Řízení displeje

Abstract

In this Bachelor thesis have been solved a problem from a sphere of development of new modules for Dominoputer kit, which is very often used for teaching subjects intent on electrical engineering. In this thesis the construction of two modules is described and it is suggested their usage.

The First processing module is the end module of washing machine. It is a simple simulator of a real washing machine. On a panel there are Led-diodes which indicate if the washing machine is turned on, door position (open or close), water level, and temperature of the water. The rotation of a barrel is indicated by eight Led-diodes arranged in a circle. These Led-diodes are controlled by the program in the chip 1016EA, which handles sequential operating of diodes. In a fact it is 8 control levels. It includes three levels of speed, each in the clock wise and anti-clock wise direction and two levels to stop the rotation of the barrel.

For writing the control program for this module and for uploading this program onto the chip have been used the program ispExpert by the company Lattice. This program uses the programming language VHDL.

The Second processing problem is a module of dot matrix LED-displays. Here is processed full and detailed description of construction and the control of communication of this display by the control circuit. The Module has been a part of a slot-machine and so one of the sources of information has become the guide-book of this machime (connection of circuit of display). Findings in this work are results of data sheet analysis and executed simulation of circuits using the simulation programs Multisim and Proteus- Isis.

Keywords: Dominoputer
Programmable logic device
VHDL
Matrix LED-display
Driving of display

Obsah

Seznam použitých symbolů a zkratek

A - Ampér, jednotka elektrického proudu, základní jednotka soustavy SI

ABEL,VHDL – programovací jazyky

CMOS – technologie výroby integrovaných obvodů

DLL a **PLL** obvody - PLL (Phase Locked Loop) nebo DLL (Delay Locked Loop) jsou obvody využívané v FPGA obvodech k obnovení charakteristik hodinového signálu

Domino – Dominoputer, modulární výukový stavebnicový systém

FPGA – Field programmable gate array, skupina programovatelných logických obvodů

IEEE 1149.1 - Standard známý jako JTAG, je jednoduchý systém pro komunikaci se zařízením a jeho vzdálené testování.

IO – Integrovaný obvod

JEDEC - Jedná se o standardní formát pro přenos informace o logickém návrhu propojek a konfigurací programovatelného obvodu z CAE programů do programátoru logických obvodů

JTAG - Joint Test Action Group je standard definovaný normou IEEE 1149.1, tzv. Standard Test Access Port (TAP). Jedná se o architekturu Boundary-Scan pro testování plošných spojů

LED - Elektroluminiscenční dioda, je elektronická polovodičová součástka obsahující přechod P-N

Log.0, Log.1 – Standardní napěťové úrovně používané pro číslicovou elektroniku.

Hz – Hertz, jednotka frekvence, základní jednotka SI

PLO – Programovatelný logický obvod

RAM – Random access memory, paměťový prvek

RTL kód - Register Transfer Language, soubor ovládacích instrukcí

V - Volt je jednotkou elektrického napětí, resp. elektrického potenciálu. V soustavě SI patří mezi odvozené jednotky, rozměr v základních jednotkách SI je: $V = m^2 \cdot kg \cdot s^{-3} \cdot A^{-1}$

Ω - Ohm je jednotka elektrického odporu, v soustavě SI patří mezi odvozené jednotky, rozměr v základních jednotkách je: $\Omega = m^2 \cdot kg \cdot s^{-3} \cdot A^{-2}$

Úvod

Podstatou zadané bakalářské práce, jak vyplývá ze zadání, je vývoj prototypů modulů navazujících na stavebnicový systém Dominoputer (dále jen Domino) firmy RC spol. s r.o.

K vytvoření vzorových úloh slouží moduly standardně dodávané ke stavebnici. Moduly jsou kvalitně zpracovány a zaručují přesnost uvedených parametrů. Systém obsahují rozličné množství ovládacích, funkčních a zobrazovacích koncových modulů. Avšak pro rozšíření možností použití stavebnicového systému je třeba výroba dalších zejména koncových modulů. Koncovým modulem rozumíme názorné zobrazovací zařízení, jakým jsou například segmentový LED-displej, nebo ve výuce oblíbená světelná křižovatka.

Po konzultaci s vedoucím bakalářské práce, jsem začal pracovat na již dříve rozpracovaném projektu koncového modulu automatické pračky. Hlavním úkolem tedy bylo oživit tento výrobek a zhotovit k němu příslušnou dokumentaci.

Druhá polovina této bakalářské práce je věnována bodovému maticovému displeji, který byl původně součástí výherního automatu. I tento displej nabízí zajímavé možnosti využití v modulární stavebnici Domino. Výsledkem této práce by měl být popis konstrukce modulu displeje a komunikace displeje s řídicím obvodem.

1 Stavebnicový systém Dominoputer

Systém stavebnice klade důraz na vysvětlení základních principů číslicové a analogové elektroniky a elektrotechniky, má skvělé didaktické vlastnosti plynoucí zejména z názornosti. Tuto modulární stavebnici tvoří otevřený systém plně kompatibilních modulů pro realizaci konkrétních aplikací z oblasti logické výstavby obvodů a systémů, které tvoří např. hardwarové vybavení počítačů a řídicích systémů.[1]

Uplatnění stavebnice Domino:

- Reálná měření při přednáškách a cvičeních
- Praktická laboratorní výuka
- Ověření výsledků výpočetních úloh měřením

Filosofií modulů stavebnice Domino je:

- Dokonalý spojovací kontakt
- Zdůraznění principu funkce
- Systém ochrany a jištění
- Funkční i fyzická kompatibilita

Moderní technologie, ochrana a přesnost jednotlivých modulů systému vede k souladu teoretické výuky s výsledky experimentu, tj. naše měření jsou „ideální“. Definovanou změnou obvodu je možno realizovat situaci, která by nastala při použití méně přesných součástek. „Reálný“ experiment pěstuje ve studentech cit pro elektroniku a vede ke schopnosti lépe využívat získané poznatky při další práci.[1]

2 Modul Pračka

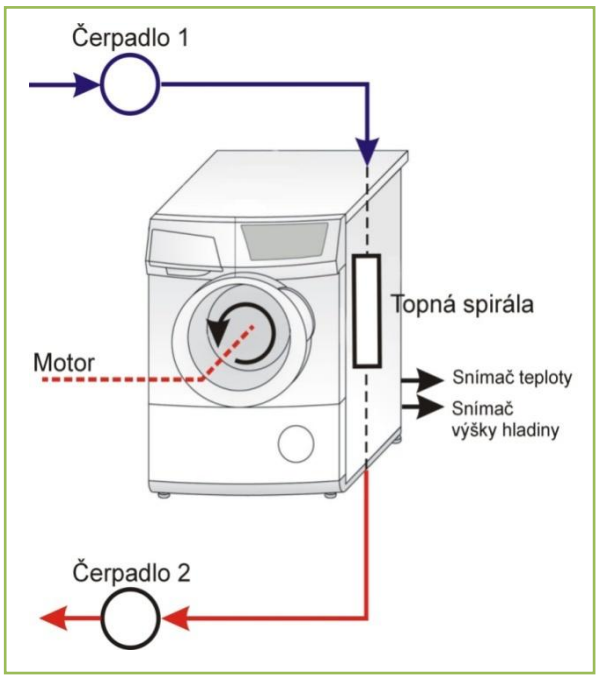
2.1 Popis modulu

Tento koncový modul simuluje praní prádla v reálné automatické pračce. A cílem úlohy využívající tento modul je naprogramovat funkční chod pračky pomocí dalších modulů ze stavebnice Domino.

Virtuální pračka (*Obr. 1*) obsahuje dvě čerpadla, první slouží k napouštění a druhé k vypouštění vody. Napouštěná voda je v bubnu ohřívána topnou spirálou. Teplota vody je monitorována snímačem teploty. Virtuální model také obsahuje snímač výšky hladiny vody v bubnu pračky. [8]

Samozřejmě pračka musí obsahovat motor, který otáčí bubnem s praným prádlem, motor se otáčí oběma směry třemi rychlostmi.

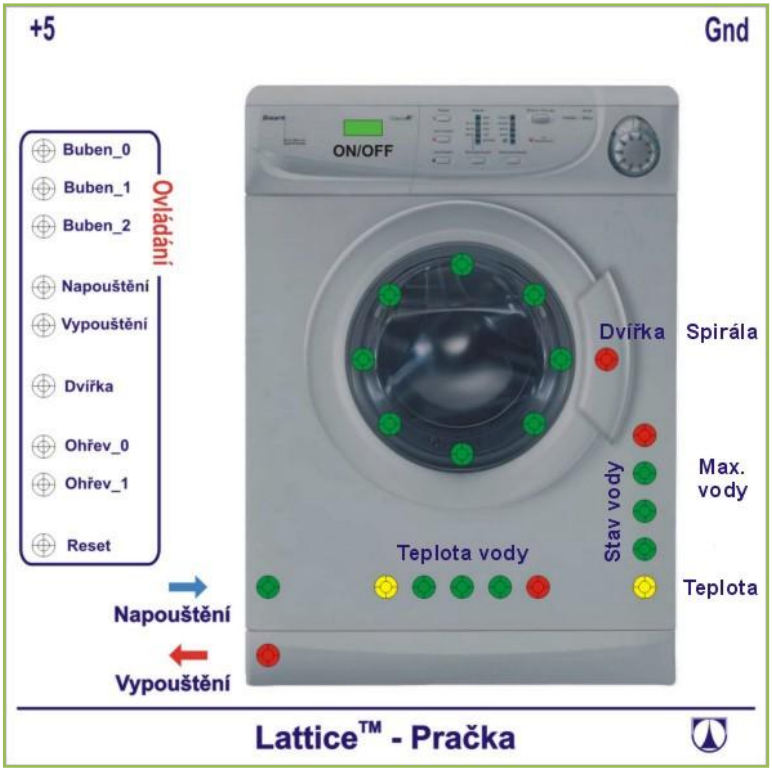
Ovládací panel pračky (*Obr. 2*) obsahuje řídící vstupy a informační výstupy, pomocí kterých lze modul řídit. Základem modulu je PLO (programovatelný logický obvod), který je řízen hodinovým signálem. Tento PLO obsluhuje všechny vstupy modulu, které vyhodnotí a zpracuje.



Obr. 1: Simulační schéma pračky

2.2 Vstupy modulu

Činnost pračky je řízená dutinkovými vstupy v ovládacím panelu, pomocí kterých lze řídit veškerou činnost virtuální pračky. Jsou zde vstupy pro ovládání otáčení bubnu, ovládání čerpadel pro napouštění a vypouštění, vstup pro koncový spínač dvířek, dále vstupy pro řízení topné spirály a vstup pro celkový reset pračky.



Obr. 2: Ovládací panel modulu [8]

2.2.1 Vstupy pro ovládání bubnu

Vhodnou kombinací vstupů (*Buben_0*, *Buben_1* a *Buben_2)* se buben roztočí požadovaným směrem s požadovanou rychlostí. Roztočení bubnu probíhá postupně, což odpovídá reálné skutečnosti, tzn. že buben se ihned neroztočí maximální rychlostí, ale začne se točit pomalu a postupně zrychluje až na maximální rychlost. Pro obsluhu točení bubnu je zavedena tato pravdivostí tabulka (*tab.1*):

Tab. 1: Pravdivostní tabulka točení bubnu

Otáčení bubnu:		B0	B1	B2
Stop		0	0	0
Doleva	Pomalu	0	0	1
	Středně	0	1	0
	Rychle	0	1	1
Stop		1	0	0
Doprava	Pomalu	1	0	1
	Středně	1	1	0
	Rychle	1	1	1

Podle pravdivostní tabulky tedy: Přivedeme-li např. na vstup *Buben* kombinaci [1 1 1] pak se buben začne roztáčet doprava a postupně se roztočí až na maximální rychlost, ve které setrvá.

2.2.2 Vstupy pro ovládání čerpadel

Jak již bylo řečeno, modul obsahuje dvě čerpadla, pomocí kterých lze regulovat výšku hladiny vody v pračce. Pomocí vstupů *Napouštění* a *Vypouštění* řídíme jednotlivá čerpadla.

Po přivedení log. 1 na vstup *Napouštění* a zároveň log. 0 na vstup *Vypouštění*, se začne zvyšovat hladina vody v pračce. To znamená, že pro spuštění čerpadla je třeba na určený vstup přiřadit log. 1.

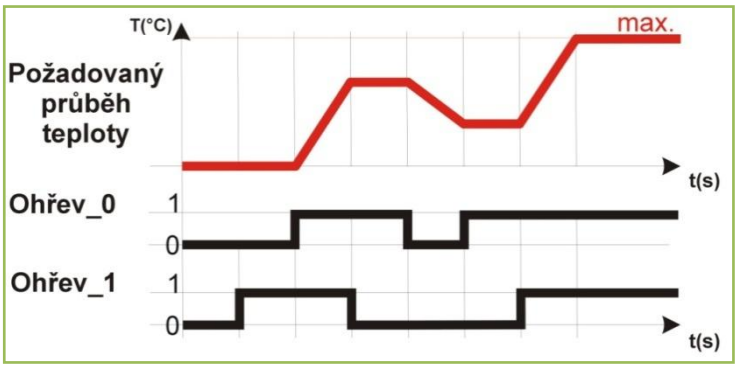
Jsou-li v chodu obě čerpadla, výška hladiny vody se nemění, protože čerpadla mají stejný sací výkon. Stav výšky hladiny vody je indikován pěti LED-diodami.

2.2.3 Vstup koncového spínače dvířek

Dále modul obsahuje vstup pro koncový spínač dvířek. Spínač je nutný, aby nedošlo k napouštění pračky, jsou-li dvířka otevřená. Pokud je na vstup přivedena log.1, pak svítí kontrolka *Dvířka* a dvířka jsou uzamčená, pokud je přivedena log. 0, pak nesvítí a dvířka jsou odblokovaná. Svítí-li tedy kontrolka, program pracího procesu by neměl dávat pokyn k napouštění.

2.2.4 Vstupy pro ohřev vody

Vhodnou kombinací logických úrovní na vstupech *Ohřev_0* a *Ohřev_1* lze dosáhnout požadované teploty. Modul rozlišuje pět úrovní teploty vody. Po dosažení maximální teploty spirály je teplota vody udržována na maximu. Příklad řízení teploty s požadovaným průběhem je uveden na obrázku (*Obr. 3*).



Obr. 3: Stavový diagram vstupů pro ohřev vody

2.2.5 Vstup reset

Celý modul lze resetovat, to znamená nastavit počáteční stavy všech hodnot, pomocí vstupu *Reset*. K resetu dojde, přivedeme-li na *Reset* jedničkový impuls. Pro normální provoz modulu je nutné, aby na vstup *Reset* byla přivedena log. 0.

2.3 Výstupy modulu

Výstupy tohoto modulu lze využít jako zdroj libovolných informací, musí k tomu být samozřejmě uzpůsoben program v PLO. V tomto případě byly využity 4 výstupy.

První výstup *Spirála* indikuje činnost spirály (topí = log. 1, netopí = log. 0). Výstup *Max. vody* indikuje plný buben vody, tedy stav kdy by mělo dojít k vypnutí čerpadla pro napouštění vody (plný buben = log. 1).

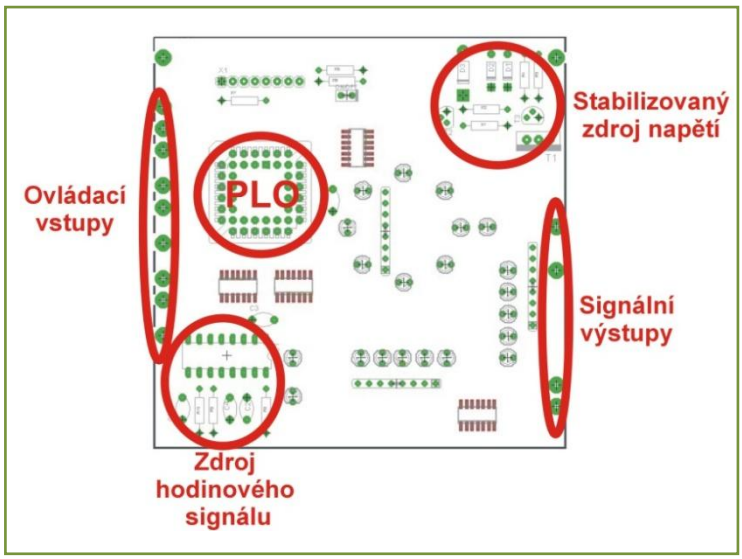
Poslední dva výstupy podávají informaci a teplotě vody. Lze jimi bohužel rozlišit pouze čtyři stavy teploty vody, pro podrobnější informace o teplotě by bylo třeba zvýšit počet těchto výstupů. Avšak teplotu 20°C a 30°C lze rozlišit pomocí výstupu *Spirála*, protože při teplotě 20°C topná spirála netopí. Pro výstupy *Teplota_0* a *Teplota_1* je opět zavedena pravdivostní tabulka (*Tab. 2*).

Tab. 2: Pravdivostní tabulka výstupů Teplota

Teplota vody [°C]	Teplota_0	Teplota_1
20	0	0
30	0	0
40	0	1
60	1	0
90	1	1

2.4 Popis konstrukce modulu

Schéma modulu je navrženo podle pravidel návrhu modulů Domino, to znamená, že obsahuje stabilizátor napětí s ochranou proti přepólování a vstupy i výstupy jsou odděleny pomocí invertorů 7404, kde invertory jednak obracejí logickou úroveň a také slouží jako jednoduchý zesilovač signálu.



Obr. 4: Popis konstrukce obvodu

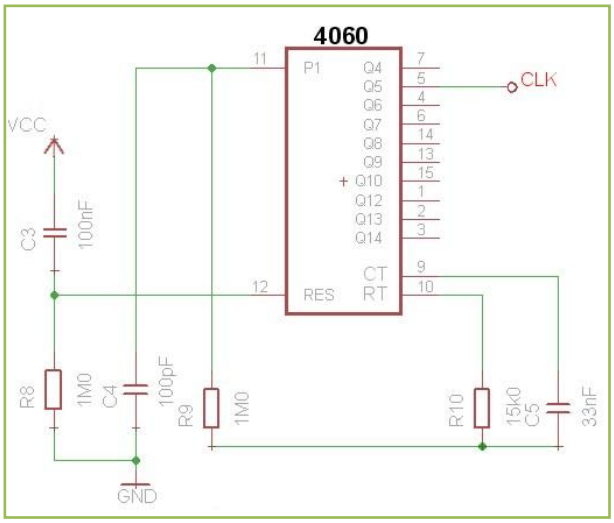
2.4.1 Popis schématu

Schéma zapojení obvodu (viz. Příloha [B]), stejně tak i deska plošného spoje (viz. Příloha [A]) jsou obsaženy v příloze.

Hlavní komponentou je PLO *ispLSI1016EA* od firmy Lattice, který je naprogramovaný pro obsluhu veškerých činností modulu. Samotný program (viz. *Příloha [E]*) je psán ve VHDL kódu. Pro programování čipu byl použit software od firmy Lattice *IspExpert*, který bude dále podrobněji popsán.

Zdroj hodinového signálu pro PLO je tvořen integrovaným obvodem *CMOS 4060* (*Obr. 5*) s příslušnými RC součástkami. Jedná se o obvod binárního čítače s děličkou kmitočtu a oscilátorem. Dle katalogových listů platí, že:

$$f_{osc} = \frac{1}{2{,}5 \times R_T \times C_T}$$



Obr. 5: Zdroj hodinového signálu

Na frekvenci tohoto signálu příliš nezáleží, musí být pouze zvolena taková úroveň, aby při zpracování programu a následném zobrazení rotace diod, při simulaci točícího se bubnu pračky, byl tento pohyb postřehnutelný pouhým okem, nebo k tomu musí být uzpůsoben alespoň řídící program. Dle uvedeného vztahu tedy platí, že frekvence hodinového signálu je při použitých součástkách cca. 800 Hz.

Jednotlivé indikační LED-diody obsažené v modulu jsou, stejně jako vstupy a výstupy modulu, odděleny od PLO pomocí invertorů 7404, které signál z PLO zesilují a logicky převracejí. Pracovní bod LED-diod je nastaven pomocí rezistorů o velikosti 270Ω.

2.5 Programování PLO

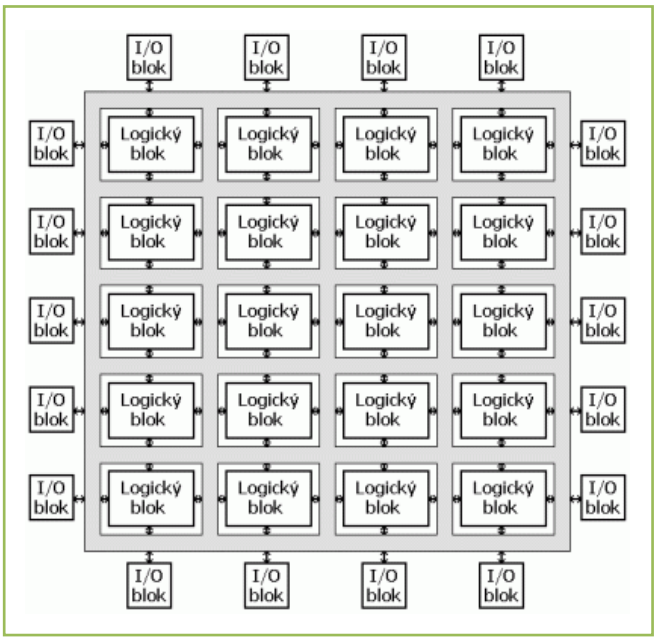
2.5.1 Programovatelný logický obvod ispLSI1016EA

Tento programovatelný obvod patří do skupiny programovatelných zakázkových obvodů (Field Programmable Logic Device). Tyto obvody lze dále dělit na:

- **PLD** (Programmable Logic Device), Simple PLD (SPLD), Erasable PLD (EPLD). Mají pevně danou strukturu typu: vstup - pole ***AND*** - pole ***OR*** - výstup
    - **PROM** (Programmable Read Only Memory)
    - **PAL** (Programmable Array Logic)
    - **GAL** (Generic Array Logic)
    - **FPLA** (Field Programmable Logic Array)
- **CPLD (Complex PLD)** Obsahují složitější architektury vycházející z PLD (vrstevnaté, s propojovací maticí, ...)
- **FPGA (Field Programmable Gate Array)** Mají pravidelnou strukturu programovatelných logických bloků s vodorovnými či svislými propojovacími linkami a propojovacími maticemi.

Použitý PLO má strukturu FPGA. Obvody FPGA - Field Programmable Gate Array jsou nejsložitějšími, zároveň však také nejobecnějšími PLD obvody. Místo klasických makrobuněk jsou složeny z tzv. logických bloků a obsahují až miliony ekvivalentních hradel (typické dvouvstupové hradlo NAND). Na obrázku (Obr. 6.) je znázorněna typická struktura obvodu FPGA.[3]

Logic Cell představují vlastní programovatelné logické bloky navzájem propojitelné globální propojovací maticí. Některé signály sousedících bloků je navíc často možné propojit přímo a bez použití globální propojovací matice. Takové spoje mají mnohem menší zpoždění a umožňují realizovat například rychlé obvody šíření přenosu, což je nezbytné například pro čítačky a násobičky. Dále FPGA obvody obsahují samozřejmě vstupně-výstupní bloky a často i řadu dalších speciálních bloků. Například to jsou hardwarové násobičky, dualportové paměti RAM, PLL a DLL obvody a další. [3],[4],[5],[7]



Obr. 6: Struktura FPGA

2.5.2 ispExpert 8.1

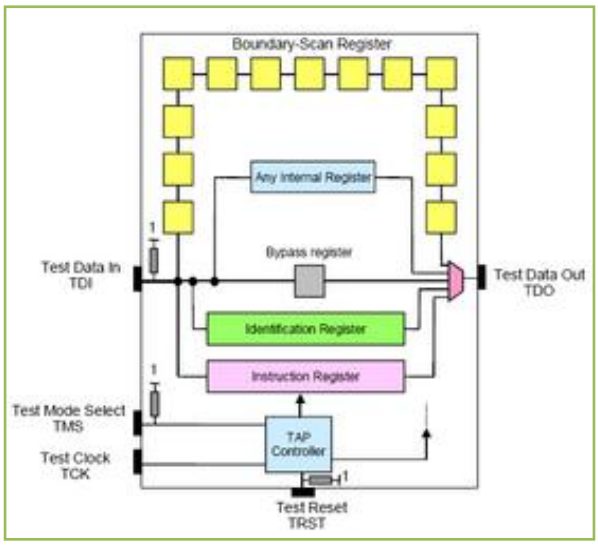
Program ispExpert je návrhové prostředí dodávané firmou Lattice Semiconductor a je určené k programování PLO *ispLSI1016EA*.

U PLO se k přenosu informace do programátoru, tj. informace o tom, jak má vypadat vnitřní struktura integrovaného obvodu, používá tzv. soubor JEDEC, označovaný příponou *.jed.

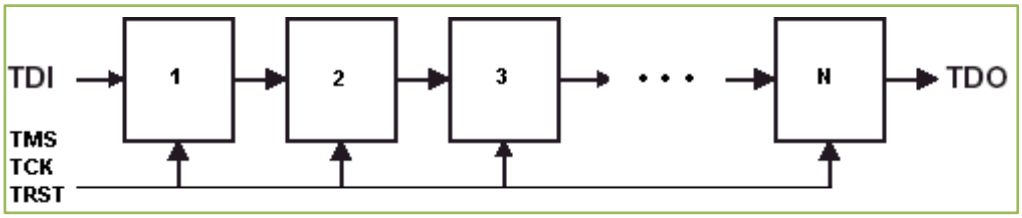
Soubor s touto příponou je jedním z výsledků návrhových systémů jakým je právě ispExpert, který tento soubor po kompilaci zdrojového souboru popisujícího vyvíjenou konstrukci vygeneruje. Soubor JEDEC svou strukturou odpovídá vnitřní struktuře PLO, která je vytvořena po nahrání souboru do PLO.

Samotné programování PLO se provádí pomocí rozhraní JTAG (Joint Test Action Group). Jedná se o standard definovaný normou IEEE 1149.1, který byl vyvinut především pro testování integrovaných obvodů, případně celých systémů. Pomocí protokolu JTAG je možné měnit stav každého (záleží na implementaci) vývodu testovaného obvodu a detekovat tak zkraty, chybné spoje, nefunkční obvod, nebo přímo ovládat připojenou součástku apod.[5]

Toto rozhraní využívá čtyři povinné signály: signál TMS(Mode), TCK(SCLK), TDI(SDI) a TDO(SDO) a nepovinný resetovací signál TRST(ispEN). TMS slouží k řízení stavového automatu, který je připojen kromě signálu TMS na signál TCK a případně TRST. Pomocí těchto signálů vytváří vnitřní signály pro řízení jednotlivých bloků uvnitř testovaného obvodu. Stav se mění každou nástupnou hranou hodinového signálu TCK. Resetovací signál je nepovinný, protože reset stavového automatu je možné provést pomocí nastavení TMS na log. 1 na dobu minimálně pěti hodinových impulsů. Je zaručeno, že ať byl automat v jakémkoliv stavu, pak po této proceduře se dostane do počátečního stavu. Podle stavu automatu je do řetězce mezi TDI a TDO připojen určitý registr. Základní povinné registry jsou instrukční registr, boundary scan registr, bypass registr a nepovinný ID registr, který slouží k detekci typu a výrobce součástky.[2], [3], [4]



Obr. 6: Struktura JTAG rozhraní



Obr. 7: Testovací řetězec

Jednotlivé testované obvody se propojí do tzv. boundary řetězce seriově pomocí TDI a TDO. Ostatní signály jsou paralelně připojeny na každou součástku. Jelikož existuje bypass registr, který pouze jakoby propojuje TDI a TDO, můžeme kdykoliv "odpojit" veškeré prvky řetězce, a ponechat aktivní pouze jeden obvod, se kterým budeme pracovat. [2],[3],[4]

Co návrhový systém ispExpert umožňuje:

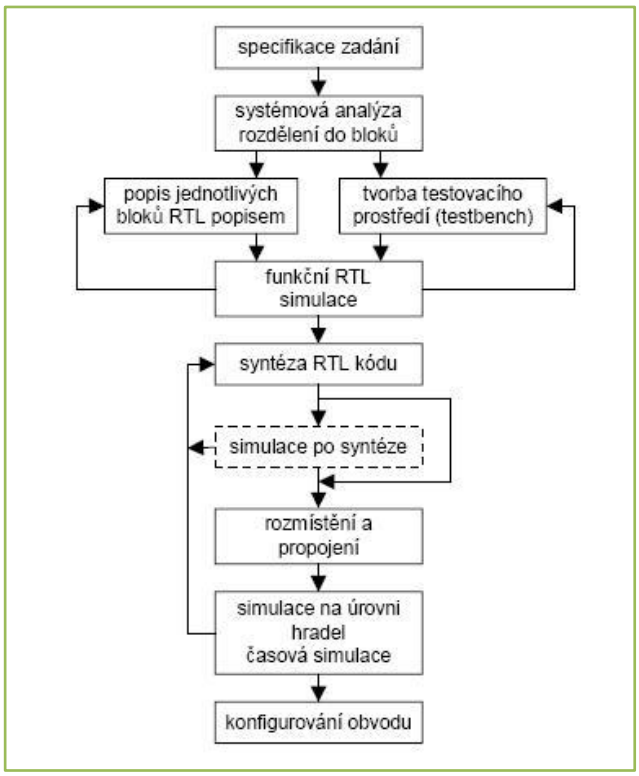
- zápis vstupních údajů o navrhovaném obvodu
- syntézu – převod zapsaných údajů do konfiguračního souboru cílového obvodu
- simulaci – ověření funkce navržených systémů
- další pomocné funkce (časová analýza – výpočet zpoždění, předstihu a přesahu apod.)

Nejpoužívanějším způsobem popisu návrhového systému je asi textový popis v jazycích HDL. Nabízí mnohem větší možnosti než při vstupu schématu, při textovém popisu lze snadno zadat atributy.

Existuje mnoho uznaných standardů textového popisu (ABEL, VHDL, Verilog…). U těchto standardů je snadná přenositelnost mezi návrhovými systémy, snadné je opětovné použití jednotlivých bloků konstrukcí (kopírováním textu). K prohlížení a editaci zdrojového textu stačí jakýkoliv textový editor. Jediná nevýhoda textového popisu je asi, že někdy může být nepřehledný a chyby se obtížně hledají. [6]

Postup při návrhu aplikace PLO:

- zápis vstupních údajů (zdrojový text, schéma…), syntaktická kontrola a oprava chyb
- funkční simulace (ověření funkčnosti návrhu bez ohledu na časové parametry)
- implementace do zvoleného cílového obvodu, tj. vytvoření obrazu požadované konfigurace PLO, makrobuněk a případně dalších programovatelných prvků
- časová simulace
- naprogramování cílového obvodu



Obr. 8: Postup při návrhu aplikace PLO

2.6 Oživení modulu

Při kontrole a proměřování obvodu modulu byly zjištěny nedostatky týkající se konstrukce obvodu. Některé součástky nebyly ideálně napájeny k desce plošného spoje, a tudíž zde nebyl žádný, nebo špatný kontakt součástek s DPS. Toto bylo nutno napravit. Dále bylo nutné vyměnit integrovaný obvod CMOS 4060 obsažený v obvodu a na vstup PLO *Y1/Reset* přivést drátovou spojkou log. 1, aby PLO nebyl ve stálém resetu.

Pro funkci obvodu je nezbytně nutné, aby PLO obsahoval program, který obsluhuje činnost modulu. Proto bylo třeba podrobně se seznámit s tímto PLO a způsoby jeho programování. Programování obvodu probíhalo ve výše popsaném prostředí ispExpert. Zdrojový kód programu (*Příloha [E]*) byl pak psán ve VHDL kódu.

2.7 Výuková úloha s modulem Pračka

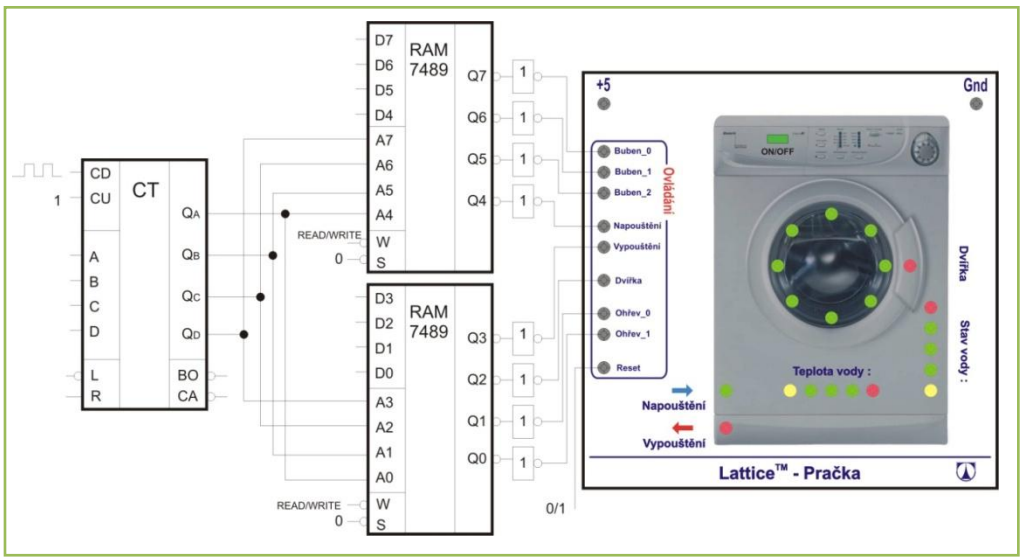
Jednou z možností jak využít tento koncový modul ve výuce je, že student naprogramuje paměť RAM 7489, kterou pak bude ovládat proces praní prádla. Účelem takovéto úlohy je tedy osvojení znalostí spojených s programováním pamětí RAM. Dané zapojení je na *Obr. 9.*

K této vzorové úloze jsou dále potřebné tyto moduly:

a) modul tlačítek
b) dva moduly s pamětí RAM (IO 7489)
c) patice s čítačem (např. IO 7493)

2.7.1 Postup programování pračky

- výstupy z tlačítkového modulu se připojí dle *Obr. 9* ke vstupům pamětí *D0 – D7* paměti se připojí na napájení a na vstupu *W* se nastaví režim zápisu
- na vstup *CD* čítače se mechanicky (připojením na log. 1) přivádí jedničkový impulz, aby čítač inkrementoval adresu, jež se přivádí na adresové vstupy pamětí
- poté se postupně naprogramuje všech 8 vstupních signálů pračky
- po ukončení programování pamětí, se paměti přepnou do režimu čtení a vstup *CD* čítače se připojí na modul *Time base* (časová základna), kde se zvolí vhodná frekvence generování hodinového signálu. Tím dojde k postupnému vyvolávání hodnot vstupů modulu *Pračka,* uložených na adresách v pamětech, potřebných k rozběhu pračky.[8]



Obr. 9: Schéma zapojení úlohy

3 Modul bodového displeje

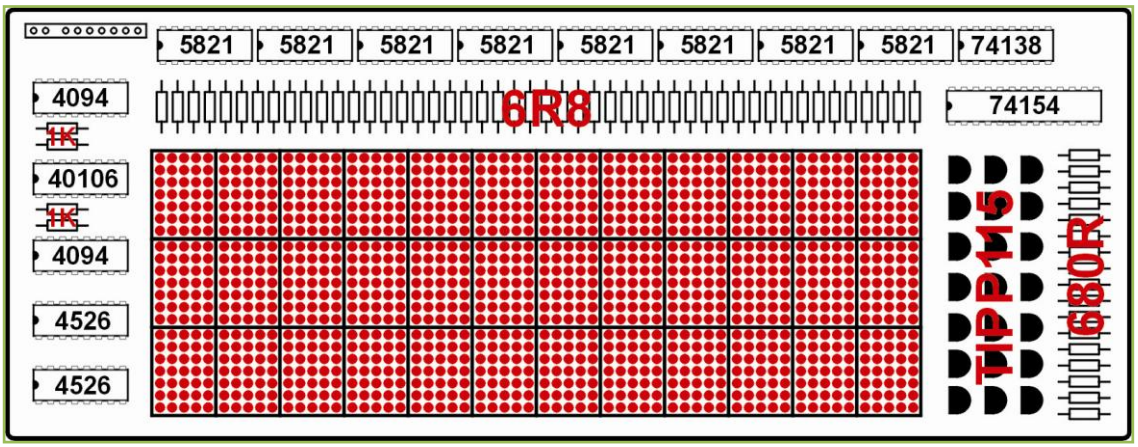
Jak již bylo v úvodu řečeno, další zpracovávanou úlohou je modul bodového maticového LED displeje (*Obr. 10*). Hlavním cílem je popsat komunikaci displeje s řídícím obvodem. Tento displej byl součástí výherního automatu firmy Ace Coin Equipment Limited a byl vyroben kolem roku 1990. Samotná deska plošného spoje nese označení *SP .ACE MATRIX DISPLAY BOARD*. V příloze je obsaženo schéma zapojení (*Příloha[D]*) i deska plošného spoje (*Příloha[C]*).

Tab. 6: Vstupní piny obvodu [9]

1.	EN DISP
2.	DATA
3.	STROBE
4.	CLOCK
5.	GND
6.	PWR GND
7.	+5.6V LOGIC
8.	(nepřipojeno)
9.	+VDISP
10.	+VDISP

3.1 Popis modulu

Displej tvoří 1260 LED-diod ve 21 řádcích a 60 sloupcích. Celý obvod je řízen deseti vstupními piny. V tabulce (*Tab.6*) jsou vypsány funkce těchto vstupních pinů.



Obr. 10: Modul LED displeje

3.2 Vstupní piny modulu

Vstup *EN DISP* je vstup pro reset celého obvodu. *DATA* je sériový datový vstup. *STROBE* je tzv. výběrový signál, při němž dojde k zobrazení požadované informace na displej. *CLOCK* je pin pro vstup hodinového signálu. Piny *GND* a *PWR GND* jsou určeny k uzemnění obvodu, *GND* pak k uzemnění logických obvodů a *PWR GND* k uzemnění samotné zobrazovací části. Přes pin *+5,6V LOGIC* jsou napájeny logické integrované obvody řídící části displeje. Pin č. 8 není připojen a poslední piny *+VDISP* jsou určeny k napájení zobrazovací části displeje, tedy jednotlivých LED-diod.

V původním zařízení byl displej, podle servisní příručky napájen (+*VDISP*) stejnosměrným napětím +5,6V/3A s tolerancí (5,3÷5,8V) o max. zvlnění 1V. Řídicí logické obvody pak stejnosměrným napětím 5,6V/2A (5,4÷5,8V) s max. zvlněním 100mV.

Napájení obvodu by se tedy mělo pohybovat velice blízko uvedeným hodnotám, avšak dle katalogových listů jednotlivých integrovaných obvodů lze předpokládat, že obvod bude funkční i při mírně nižší hodnotě napájecího napětí (+5V) což odpovídá standardům stavebnice Domino.[9]

3.3 Popis konstrukce obvodu displeje

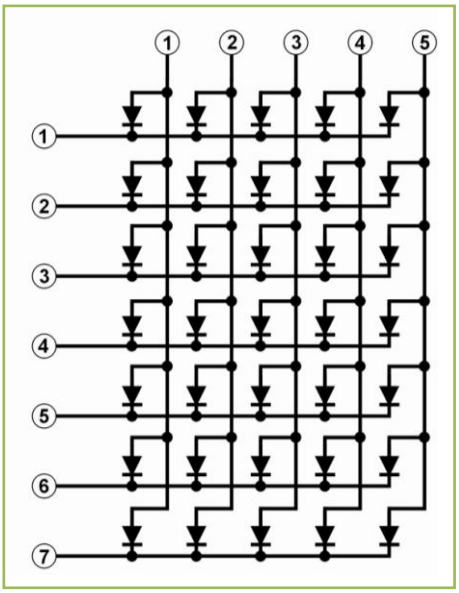
Obvod lze rozdělit do dvou základních segmentů, tj. řídicí a zobrazovací část displeje. Zobrazovací část tvoří soustava 36-ti bodových displejů 5x7 LED-diod (*Obr. 11*). Celá soustava 1260 LED-diod je tedy spojena maticově.

3.3.1 Zobrazovací část displeje

Funkce maticových displejů je velice jednoduchá. Po připojení kladného napětí na zvolenou anodu a uzemnění příslušné katody se rozsvítí požadovaná LED-dioda.

Například při požadavku na rozsvícení poslední diody (5) v prvním řádku displeje (1) je nutné připojit napájení k pinům (5,1) = (sloupec, řádek).

Pracovní body jednotlivých LED-diod jsou nastaveny na cca. 2,1V/10mA maximálně však až 5V. Svítí-li tedy větší množství diod, je třeba počítat s nárůstem proudu.



Obr. 11: Vnitřní zapojení LED displeje LTP-757

Pro rozsvícení celého sloupce displeje, tedy 21 LED-diod, je třeba cca 210mA. Což by teoreticky při rozsvícení celého displeje najednou, mohlo činit i několik ampér. Jelikož obvod neobsahuje žádné výkonové členy, je patrné, že jednotlivé diody (resp. řádky displeje) nejsou rozsvěceny všechny zároveň, ale jsou rozsvěceny po velice krátkou dobu v řádech milisekund jeden po druhém, což umožňují použité tranzistory v darlingtonově zapojení (*TIPP115*). Tímto způsobem lze získat po velice krátkou dobu vysokých hodnot napájecího proudu až v řádech jednotek ampér.

Z předchozích závěrů tedy plyne, že všechny diody nesvítí zároveň, nýbrž jsou jako většina displejů ovládány vysokou frekvencí po řádcích. Při dlouhém intervalu přivedeného proudu by došlo ke zničení proudových zesilovačů *TIPP115*. Délkou nebo počtem proudových impulsů lze pak samozřejmě docílit i libovolného jasu displeje.

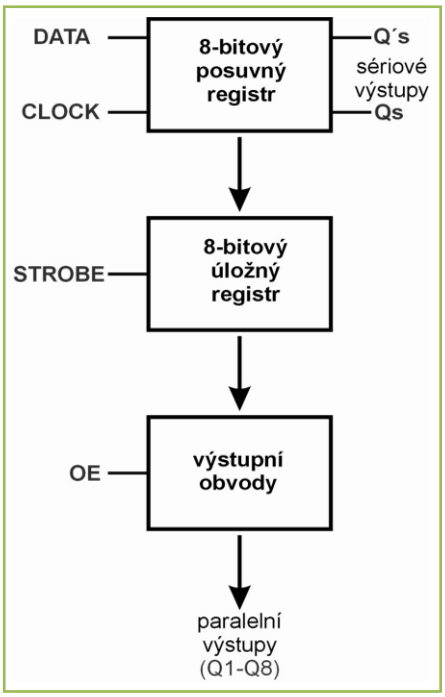
Displej je tedy nutno řídit vysokou frekvencí, protože jinak by docházelo k blikání řádků displeje. Blikají-li jednotlivé řádky vysokou frekvencí, lidské oko to nepostřehne. V praxi se při řízení podobných displejů osvědčilo obsluhovat nejprve např. sudé řádky a poté liché. Blikání displeje není tak markantní a dovoluje tedy použít i menší frekvenci řízení displeje.

Displej je tedy ideální řídit pomocí mikropočítače nebo pomocí PLO použitého v předešlé úloze modulu *Pračka*, samozřejmě s použitím paměti, kde by byly jednotlivé zobrazované uloženy.

3.3.2 Řídicí část displeje

Řídicí část displeje je možné rozdělit na další tři segmenty. A to sice část pro volbu řádků, část pro volbu sloupců a část řízení proudového napájecího impulsu displeje, tedy část řízení jasu displeje.

Všechny řídicí vstupy displeje jsou odděleny pomocí invertorů (*IC2*) *HEF40106BP*, které plní také funkci zesilovače. Z charakteru zapojení plyne, je-li na řídící vstup přivedena log.0, na výstupu invertoru je log. 1. Ovšem mimo vstup *EN DISP* neboli reset, tady jsou zařazeny dva invertory za sebou.



Obr. 12: Posuvný registr HCF4094BE

Řízení jasu displeje

Sériový datový signál vstupuje nejprve do části obvodu, kde je řízen jas displeje. Integrovaný obvod IC3 *HCF4094BE* je 8-bitový posuvný a úložný sběrnicový registr (*Obr. 12*). Data se posouvají při náběžné hraně hodinového signálu. Je-li vstup *STROBE* v log. 1, jsou data z každého bitu posuvného registru převedena do paměťového registru. Data z paměťového registru se objeví na paralelních výstupech obvodu je-li na vstupu *OE* log. 1. Protože je vstupu *OE* stále přiřazena log.1, data se objevují na paralelních výstupech IO ihned po příchodu signálu *STROBE*.

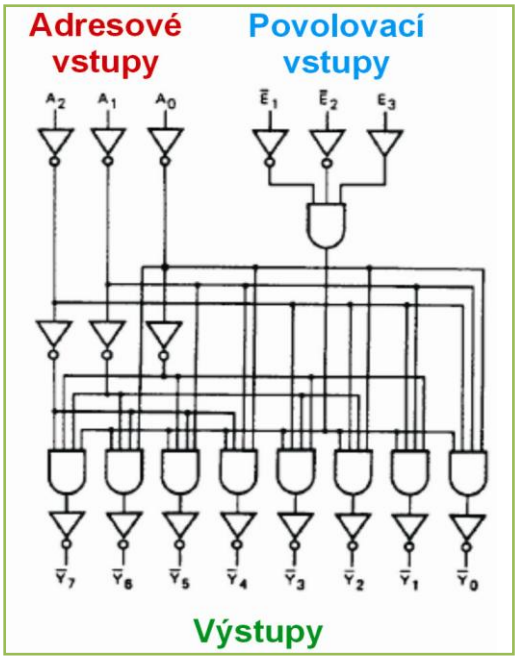
Získaná 8-bitová paralelní informace se dále zpracovává v (*IC4* a *IC5*) *HEF4526BE* kaskádně zařazených programovatelných 4-bitových binárních sestupných čítačích. Kaskádní řazení zde vytváří rozšíření, umožňující zpracování dat o potřebné šířce 8-bitů, což je 256 možných stavů. Tyto sestupné čítače pracují vlastně ve funkci časovače a po příchodu signálu *STROBE* je paralelní informace získaná z posuvného registru naprogramována do čítačů a následuje odpočítávání. Vždy když výstup *Q4* čítače *IC4*, který je taktován signálem *CLOCK*, přechází z log.0 na log.1, uděluje čítači *IC5* vzestupnou hranu hodinového signálu potřebnou pro odpočet jednoho stavu čítače *IC5*.

Dalším důležitým stavem IO *IC4* je nulový stav, kdy se na výstupu *TC* („0“) objevuje log. 1. Tento signál je dále veden přes invertor do dekodéru *IC14*, kde „povoluje“ zobrazení adresy řádku o které bude hovořeno později. Tímto způsobem je udělen proudový impuls o délce cca. jedné periody hodinového signálu a je rozsvícen požadovaný počet LED-diod v řádku displeje. Počet těchto proudových impulsů je pak úměrný původní paralelní informaci z IO *IC3*. Tzn. že čím vyšší binární hodnota bude uložena v těchto 8-bitech, tím více proudových impulsů bude prostupovat LED-diodami, tudíž celkový světelný výkon LED-diod bude větší.

Volba řádku

Sériový signál postupuje dále přes IO *IC3* z části řízení jasu beze změn do dalšího posuvného a úložného sběrnicového registru (*IC1*), ten již patří do další části obvodu, kterou je část pro volbu řádku displeje.

Posuvný registr opět při příchodu signálu STROBE převede sériovou informaci na paralelní o šířce 8-bitů. Protože na displeji je pouze 21 řádků, není tedy využito všech 8-bitů, ale postačuje jich pouze 5. To dává možnost rozlišit 32 stavů (adres řádků). Tato paralelní informace o šířce 5-bitů je předána do kaskádně zapojených dekodérů *IC14* (*74HC138*) a *IC15*(*M74HC154*).



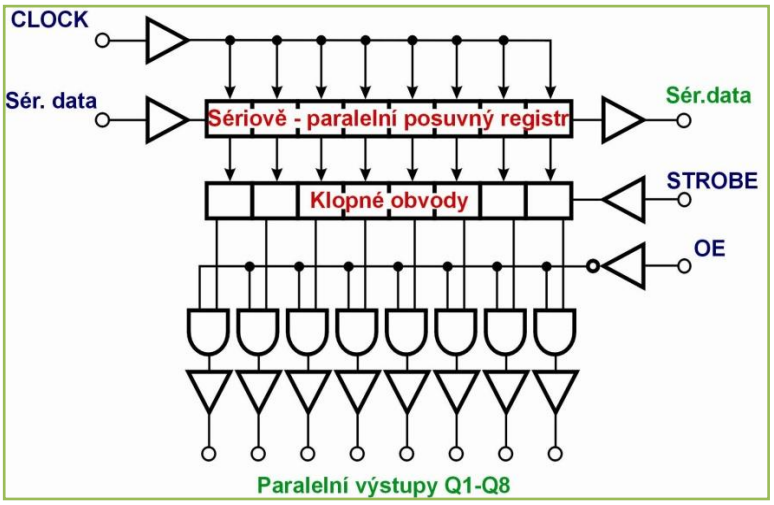
Obr. 13: Logické zapojení dekodéru 74HC138

Dekodéry mají obdobnou funkci, liší se pouze šířkou zpracovávaných dat. (*Obr. 13*) Jednoduše popsaná funkce dekodérů by mohla znít asi takto. Dekodéry podle vstupů, kam jsou přivedeny adresy výstupů, určují který z výstupů bude v log. 0, ostatní zůstávají v log. 1 což je výchozí stav. K nastavení výstupů dojde pouze, je-li na povolovací vstupy dekodérů přivedena vhodná kombinace log. úrovní. Povolovacím impulsem je zde signál z časovačů (*IC4* a *IC5*), o němž bylo hovořeno v části řízení jasu displeje. Samozřejmě předtím musí přijít i spínací signál *STROBE*, aby byla vůbec získána adresa řádku.

Je-li na jednom z jednadvaceti připojených výstupů dekodérů log. 0, dojde k otevření báze příslušného tranzistoru v darlingtonově zapojení (*TIPP115*) a může tak začít procházet proud do zvoleného řádku displeje. Proud má díky ovládacímu signálu z části obvodu regulace jasu impulsový charakter, který způsobuje povolování či zakazování zobrazení výstupů na dekodérech v závislosti na stavech časovačů.

Volba sloupce

Posledním úsekem obvodu kam vstupuje řídící sériový signál je soustava osmi sériově řazených posuvných registrů *UCN 5821A* (*IC6-IC13*), které převádí sériový signál na paralelní, to se děje opět při spouštěcím signálu *STROBE*. Tato část řídícího obvodu pro volbu sloupce displeje vyžaduje k přesnému určení adresy řádku, který má být „aktivní“, informaci o šířce 60-bitů. Na stavu posledních čtyř výstupů posuvného registru (*IC13*) nezáleží, protože nejsou použity.



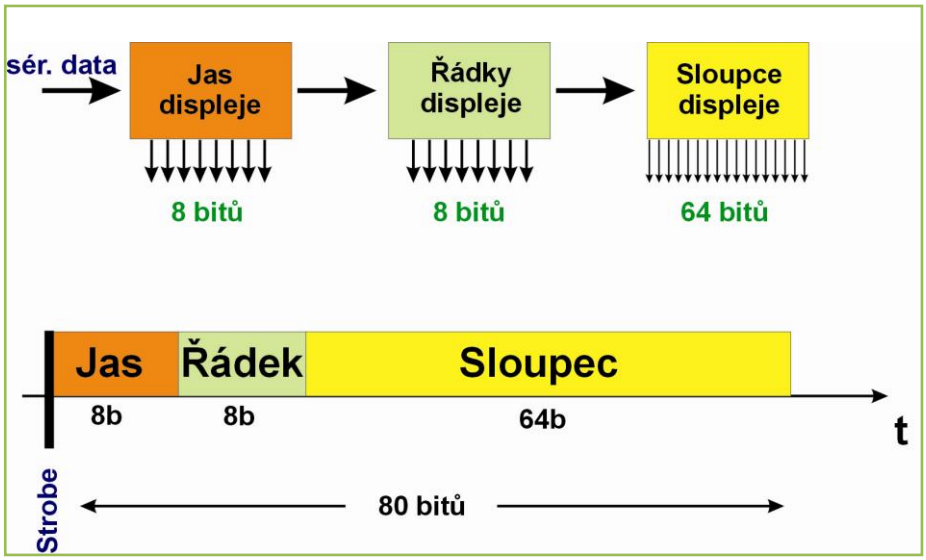
Obr. 14: Logické schéma posuvného registru 5821A

Po „napumpování“ 64-bitové sériové informace o adrese sloupce, přichází opět spouštěcí signál *STROBE*, ten promítne příslušné bity na výstupy posuvných registrů, ovšem za předpokladu, že vstup *OE* (připojený na celkový reset obvodu) je v log. 0.
Aby došlo k rozsvícení požadované diody v displeji, musí na příslušný řádek displeje přicházet proud výše popsaným způsobem a na příslušný sloupec musí z daného výstupu posuvného registru (*IC6-IC13*) být přiváděn signál o úrovni log. 0.

Ještě je nutné podotknout, že pomocí rezistorů R1-R21 o velikosti 680Ω a rezistorů R29-R82 o velikosti 6,8Ω, je nastaven (omezen) proud protékající jednotlivými LED diodami.

3.4 Řízení displeje

Z popsané konstrukce displeje plyne, že veškerá komunikace s displejem je sériová, protože jednotlivé úseky řídicí části displeje jsou také řazeny sériově. Data postupují obvodem synchronně s hodinovým signálem CLOCK.



Obr. 15: Datový tok pro řízení displeje

Pro řízení jednoho řádku displeje musí být tedy data do sériového datového vstupu „napumpována“ následovně. Jako první vstupuje 64 bitů popisujících adresu sloupce, přičemž na posledních 4 bitech nezáleží, protože nejsou konstrukčně využity. Ihned poté následuje 8 bitů, které určují adresu řádku (0X0=adresa poslední diody vpravo ve spodní řadě). Na prvních 3 bitech opět nezáleží, protože nejsou také využity. Posledním 8-bitovým slovem je informace o úrovni jasu displeje (nulová hodnota=100% jasu).

Po této 80-bitové posloupnosti dat, musí nutně následovat spouštěcí signál *STROBE*, ten způsobí načtení dat do řídících obvodů. Samozřejmě vše se musí dít při vstupu hodinového signálu do řídícího obvodu. Dle informací by měl být plně dostačující hodinový signál *CLOCK* o frekvenci 8MHz. Řídící signál *RESET* umožňuje vynulovat celý systém obvodu modulu displeje.

Ihned po příchodu signálu *Strobe* můžeme pokračovat s nahráváním dat pro další řádek. Jak již bylo řečeno, optimální je zobazovat informace například nejprve v sudých a poté v lichých řádcích kvůli blikání displeje. Při cyklickém řízení displeje tak není blikání pouhým okem rozeznatelné.

3.5 Výuková úloha s modulem displeje

Cílem výukové úlohy, která by tento prvek využívala opět jako koncový modul, může například být programování mikropočítačů nebo jiných programovatelných obvodů. Úkolem studentů tak může být programování např. PLO, použitého při konstrukci modulu *Pračka* v první polovině této bakalářské práce, a seznámení se s vývojovým prostředím ispExpert nebo s novější verzí ispLever. Může tak být také využit k osvojení programování ve VHDL nebo ABEL jazycích.

Závěr

Cílem mé bakalářské práce byl vývoj modulů stavebnice Domino, které by vhodným způsobem doplnily výuku.

Koncový modul *Pračka* zpracovávaný v první části této práce byl podrobně popsán, úspěšně oživen a programovatelný logický obvod FPGA byl naprogramován podle požadavků na funkci tohoto modulu, tj. simulaci praní prádla ve skutečné automatické pračce. Tento koncový modul se samozřejmě rozsahem funkcí nevyrovná reálné automatické pračce, ale pro názorný příklad byly vybrány pouze ty nejdůležitější funkce potřebné k činnosti pračky.

Abych mohl naprogramovat použitý PLO v modulu *Pračka*, musel jsem se podrobně seznámit s návrhovým prostředím ispExpert od firmy Lattice a s programovacím jazykem VHDL, ve kterém jsem psal příslušný program.

Druhá část této práce byla věnována podrobnému popisu maticového bodového LED-displeje, zejména pak jeho řízení. Bylo také nastíněno využití tohoto modulu pro výuku.

Jednotlivé části obvodu byly podrobně popsány a z nich pak byl odvozen postup pro řízení tohoto displeje. Pro pochopení funkce a řízení displeje jsem využil několik programů simulujících elektronické obvody. Jedním takový byl například Multisim 10 od firmy National Instruments. Převážná práce však spočívala v nastudování funkcí jednotlivých integrovaných obvodů přímo z katalogových listů těchto součástek.

Použitá literatura

[1] RC spol. s r. o. přístroje pro vědu a vzdělání [online].[cit. 10.12.2008]
URL: < http://www.rcdidactic.cz/cz/>.

[2] VAŠÍČEK, Zdeněk: *VHDL prakticky.* Vysoké učení technické v Brně, fakulta informačních technologií. [online].[cit. 10.12.2008]
URL: <http://www.stud.fit.vutbr.cz/~xvasic11/cl.cpld/index.php>.

[3] Pandatron.cz, elektrotechnický magazín: *CPLD a FPGA* [online].[cit. 11. 2.2009]
URL: <http://pandatron.cz/?481&cpld_a_fpga_1.dil_-_predstaveni_obvodu>

[4] NOVÁK,Ondřej: *Přednášky z předmětu Číslicová elektronika*.
Fakulta mechatroniky, Technická univerzita v Liberci, 2008.

[5] POUPA, Martin: *Přednášky z předmětu Programovatelné logické obvody*.
Fakulta elektrotechnická, Západočeská univerzita v Plzni [online].[cit. 11. 2.2009]
URL: <http://www.vyuka.fel.zcu.cz/kae/plo/plo_prednasky.pdf>

[6] Lattice Semiconductors corporation: *ispEXPERT Compiler Getting Started Manual, ispEXPERT Compiler User Manual, ISP Daisy Chain Download User Manual* [online].[cit. 11.12.2008]
URL: <http://www.latticesemi.com/dynamic/index.cfm?fuseaction=view_category&document_type=35&source=topnav >

[7] KOLOUCH, J.: *Programovatelné logické obvody a návrh jejich aplikací v jazyku VHDL*.Skriptum FEKT VUT v Brně. Brno, 2006

[8] SKALA, NEDVÍDEK: *Návrh a výroba výukových modulů Domino*.
Ročníkový projekt. Fakulta mechatroniky, Technická univerzita v Liberci, 2005.

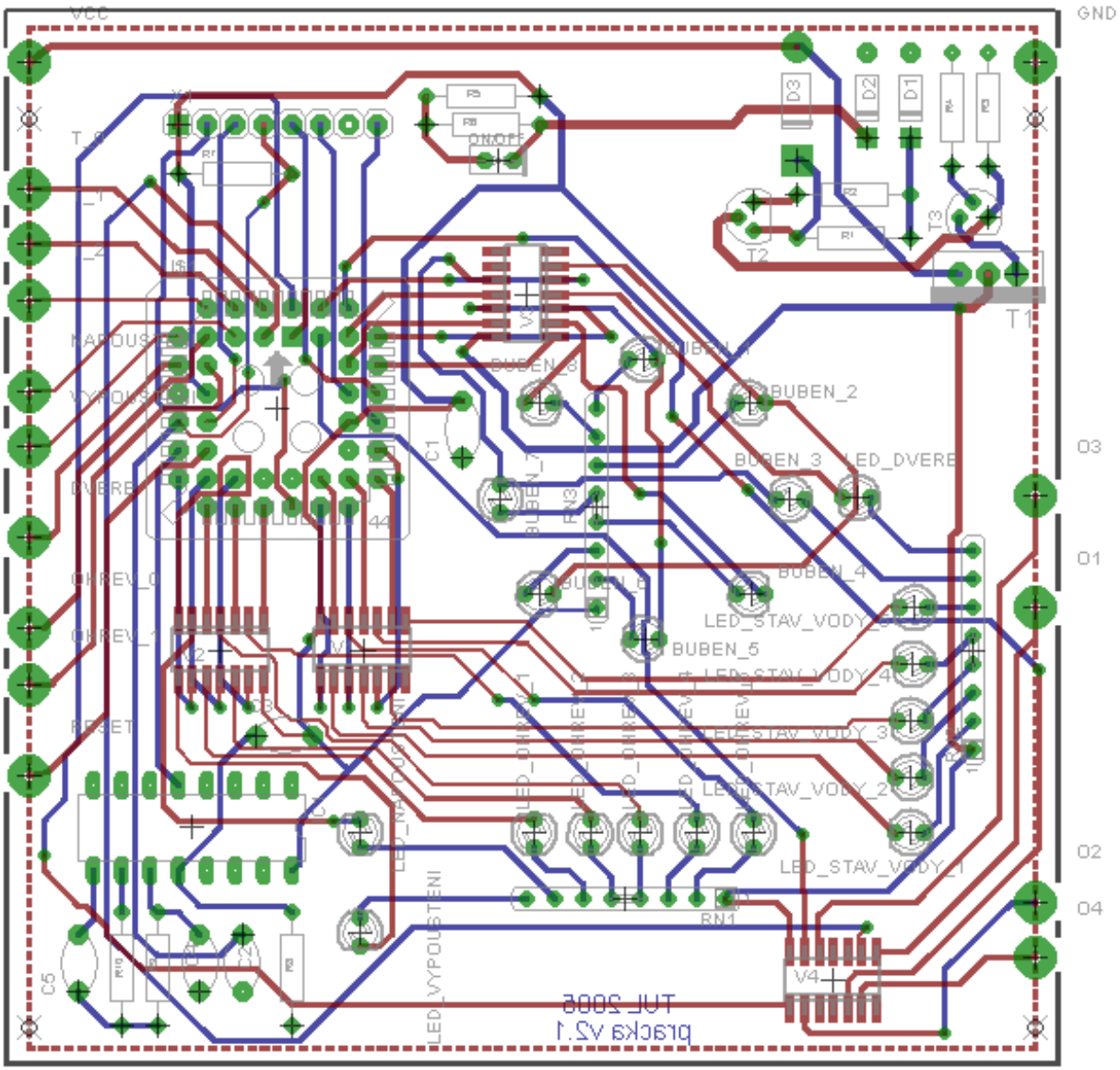
[9] Ace Coin Equipment Limited: *SP.ACE system servis manual*. West Midlands, USA, April 1989.

Struktura přiloženého DVD

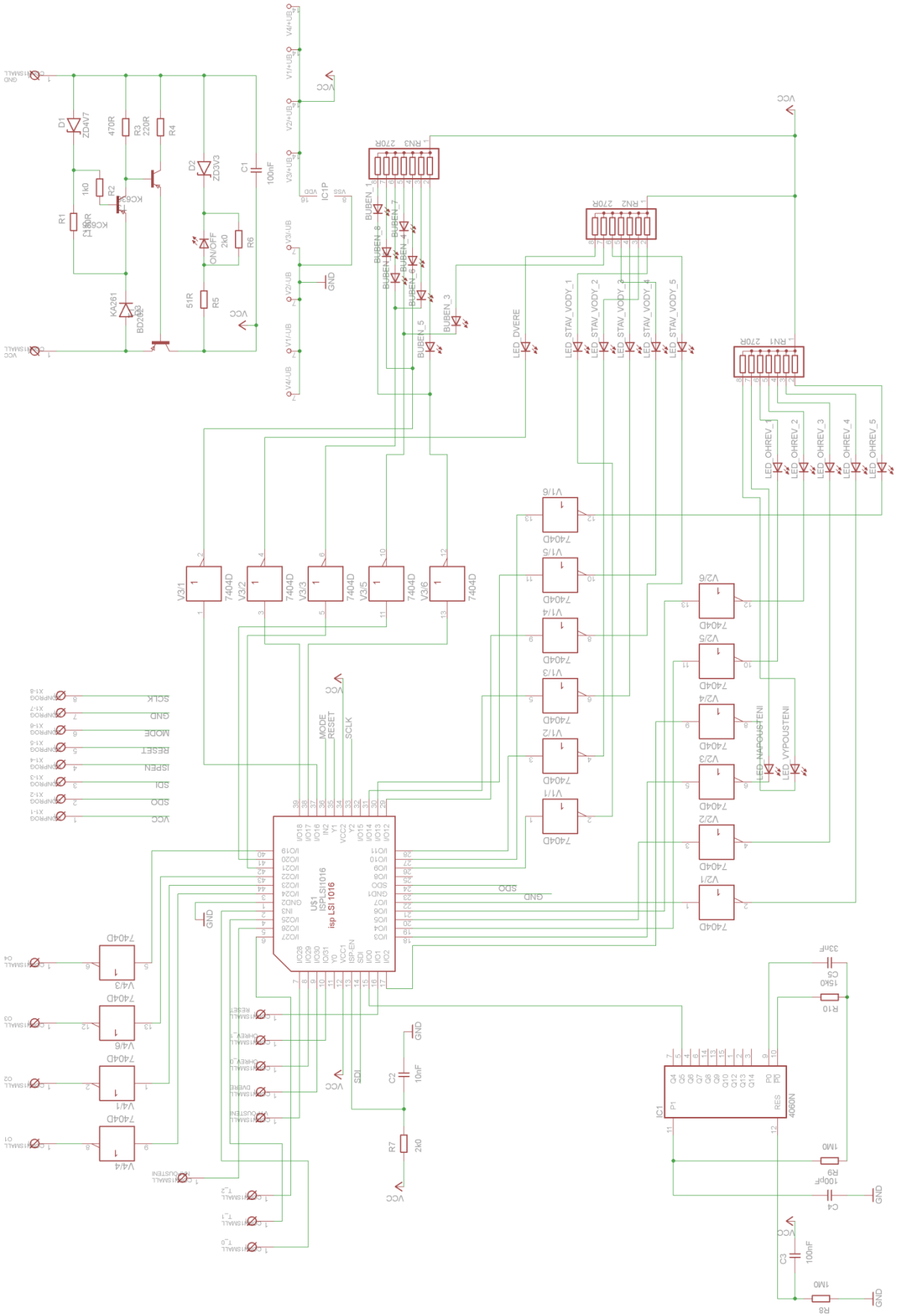
1. Složka s bakalářskou prací ve formátu .pdf .

2. Složka s deskami plošných spojů, schématy zapojení a VHDL kódem programu obsaženého v PLO modulu *Pračka*.

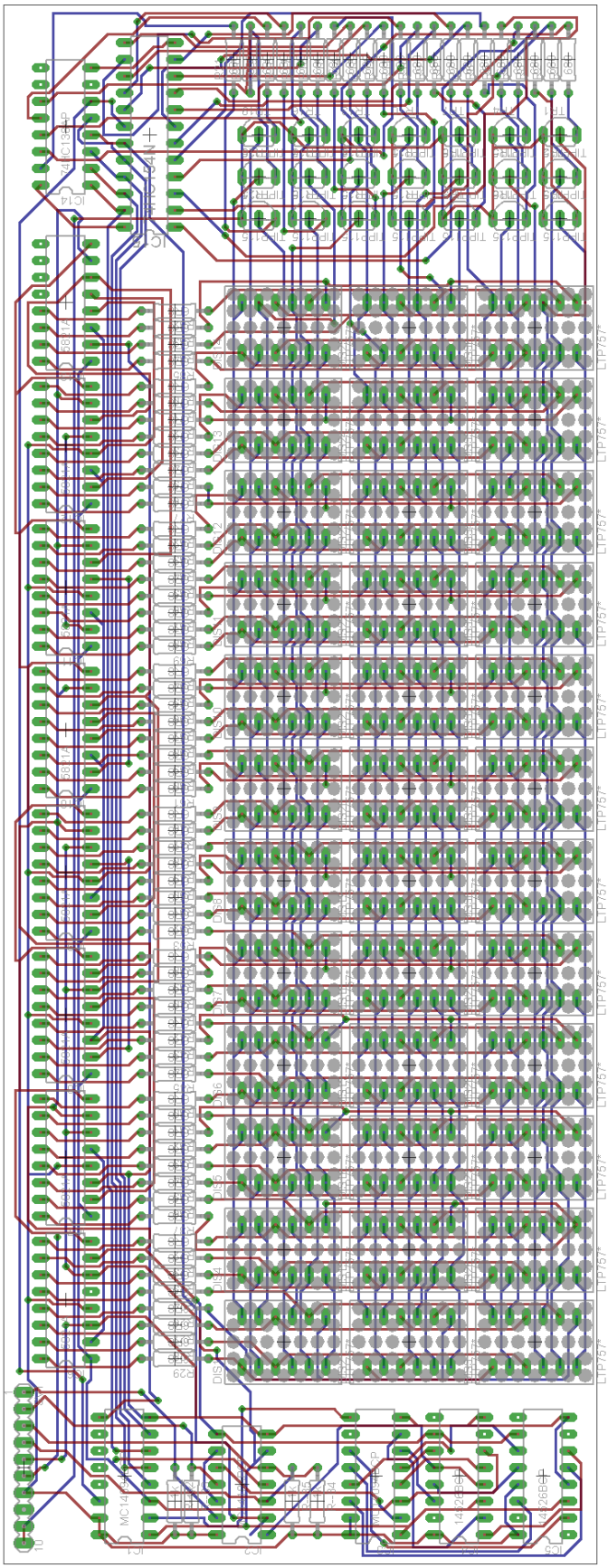
Příloha (A): Deska plošného spoje modulu Pračka



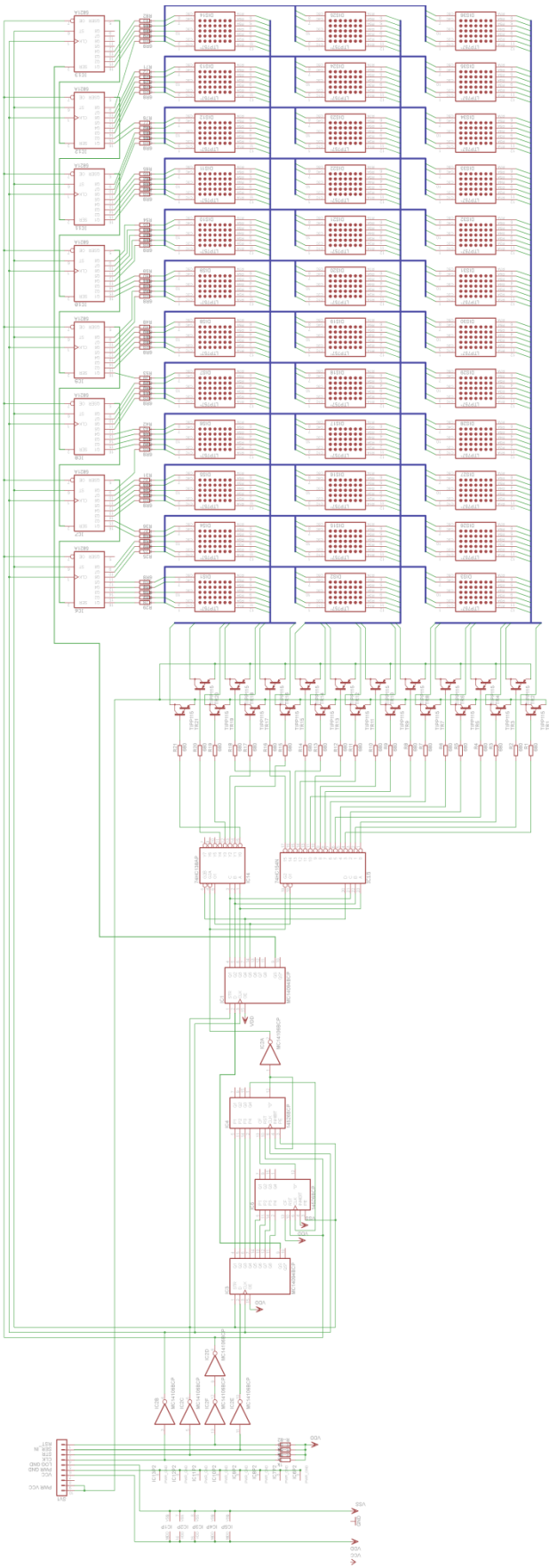
Příloha (B): Elektronické schéma obvodu modulu Pračka



Příloha (C): Deska plošného spoje modulu Displej



Příloha (D): Schéma obvodu modulu Displej



Příloha (E): VHDL kód programu

```
library ieee;
use ieee.std_logic_1164.all;

entity pracka is
PORT(   -- HODINOVY SIGNAL + RESET --
        clk, reset                      : IN    BIT;                    -- vstup indikujici zavrena dvirka
        -- VSTUPY -- (dutinky)
        Dvirka                          : IN    BIT;
        Cerpadla                        : IN    BIT_vector (1 DOWNTO 0); -- 1. cerpadlo = Napousteni, 2. = Vypousteni
        AdresaProOtaceni                : IN    BIT_vector (2 DOWNTO 0); -- adresa pro vyber smeru a rychosti otaceni bubnu
        ProudSpiralou                   : IN    BIT_vector (1 DOWNTO 0);
                                                -- 1. bit: "1 = teplota stoupa, 0 = Teplota je na konst. hodnote"
                                                -- 2. bit: "1 = Teplota je na konst. hodnote, 0 = Teplota pomalu klesa"
        -- VYSTUPY -- (rozsviceni diod)
        DvirkaDioda                     : OUT   BIT;
        Plno                            : OUT   BIT;
        netopi                          : OUT   BIT;
        Teplota                         : OUT   BIT_vector (1 DOWNTO 0);
        StavVodyDiody, SpiralaDiody     : OUT   BIT_vector (4 DOWNTO 0);
        NapousteniVypousteni            : OUT   BIT_vector (1 DOWNTO 0); -- diody ukazujici Napousteni a Vypousteni
        BubenDiody                      : OUT   BIT_vector (3 DOWNTO 0));-- ve skutecnosti bude diod 8
                                                                        -- BubenDiody(0) = 0. a 4. dioda
[paralelne]
                                                                        -- BubenDiody(1) = 1. a 5. dioda
                                                                        -- BubenDiody(2) = 2. a 6. dioda
                                                                        -- BubenDiody(3) = 3. a 7. dioda
 end;
architecture behavioral of pracka is
TYPE StavyVodyVBubnu        IS (SV1, SV2, SV3, SV4, SV5);       -- Stav Vody
        TYPE CinnostSpiraly IS (SS1, SS2, SS3, SS4, SS5);       -- Stav Spiraly
        TYPE ToceniBubnu    IS (N, L1, L2, L3, R1, R2, R3);     -- Stavy otaceni N=Netoci, L=Toci vlevo, R=Toci vpravo

        SIGNAL StavVody     : StavyVodyVBubnu;
        SIGNAL Spirala      : CinnostSpiraly;
        SIGNAL StavToceni   : ToceniBubnu;

        -- Rychlost otáčení bubnu (Zadavana pomoci adresoveho vstupu AdresaProOtaceni):
        --                              T0  T1  T2
        --                              ------------
        --          NETOCIT             0   0   0
        --          DOLEVA  Pomalu      0   0   1
        --                  Stredne     0   1   0
        --                  Rychle      0   1   1
        --          DOPRAVA Pomalu      1   0   1
        --                  Stredne     1   1   0
        --                  Rychle      1   1   1

        SIGNAL PomocnaToceni        : BIT_vector  (3 DOWNTO 0);
        SIGNAL Zobrazit             : BIT;
        SIGNAL ZmenitStav           : BIT;
        SIGNAL Krok1                : integer range 0 to 8;
        SIGNAL Krok2                : integer range 0 to 8;
        SIGNAL KrokBuben            : integer range 0 to 16;
        SIGNAL KrokOhrev            : integer range 0 to 24;

BEGIN

-- ____________________________________________________________________________________________
--
--              PROCES PRO ZOBRAZENI STAVU VODY V BUBNU
-- ____________________________________________________________________________________________

ZobrazitStavyVody: PROCESS (StavVody)
BEGIN
        CASE StavVody IS
                WHEN SV5 => StavVodyDiody <= "11111";
                            Plno <= '0';
                WHEN SV4 => StavVodyDiody <= "01111";
                            Plno <= '1';
                WHEN SV3 => StavVodyDiody <= "00111";
                            Plno <= '1';
                WHEN SV2 => StavVodyDiody <= "00011";
                            Plno <= '1';
                WHEN SV1 => StavVodyDiody <= "00001";
                            Plno <= '1';
        END CASE;

END PROCESS ZobrazitStavyVody;
```

```
-- ____________________________________________________________________________
--
--                  PROCES PRO ZOBRAZENI STAVU OHREVU SPIRALY
-- ____________________________________________________________________________

ZobrazitStavyTeploty: PROCESS (Spirala)
BEGIN

        CASE Spirala IS
                WHEN SS5 => SpiralaDiody <= "11111";
                        teplota <="00";
                        netopi <='0';           -- jestlize spirala topi, na vystupu je 1
                WHEN SS4 => SpiralaDiody <= "01111";
                        teplota <="01";
                        netopi <='0';
                WHEN SS3 => SpiralaDiody <= "00111";
                        teplota <="10";
                        netopi <='0';
                WHEN SS2 => SpiralaDiody <= "00011";
                        teplota <="11";
                        netopi <='0';
                WHEN SS1 => SpiralaDiody <= "00001";
                        teplota <="11";
                        netopi <='1';
        END CASE;

END PROCESS ZobrazitStavyTeploty;
-- ____________________________________________________________________________
--
--                  PROCES PRO ZMENU STAVU TOCENI BUBNU
-- ____________________________________________________________________________

ZmenyStavu: PROCESS (ZmenitStav ,RESET)
BEGIN
                IF (reset = '1') THEN                   -- Pri resetu se nastaví stav toceni na N (Netoceni)
                        StavToceni <= N;

                ELSIF (ZmenitStav'event and ZmenitStav = '1') THEN

                        CASE StavToceni IS
                                WHEN L3 =>
                                        CASE AdresaProOtaceni IS
                                                WHEN "011" => StavToceni <= L3;
                                                WHEN "010" => StavToceni <= L2;
                                                WHEN "001" => StavToceni <= L2;
                                                WHEN "000" => StavToceni <= L2;
                                                WHEN "101" => StavToceni <= L2;
                                                WHEN "110" => StavToceni <= L2;
                                                WHEN "111" => StavToceni <= L2;
                                                WHEN OTHERS=> StavToceni <= L2;
                                        END CASE;
                                WHEN L2  =>
                                        CASE AdresaProOtaceni IS
                                                WHEN "011" => StavToceni <= L3;

                                                WHEN "010" => StavToceni <= L2;
                                                WHEN "001" => StavToceni <= L1;
                                                WHEN "000" => StavToceni <= L1;
                                                WHEN "101" => StavToceni <= L1;
                                                WHEN "110" => StavToceni <= L1;
                                                WHEN "111" => StavToceni <= L1;
                                                WHEN OTHERS=> StavToceni <= L1;
                                        END CASE;
                                WHEN L1  =>
                                        CASE AdresaProOtaceni IS
                                                WHEN "011" => StavToceni <= L2;
                                                WHEN "010" => StavToceni <= L2;
                                                WHEN "001" => StavToceni <= L1;
                                                WHEN "000" => StavToceni <= N;
                                                WHEN "101" => StavToceni <= N;
                                                WHEN "110" => StavToceni <= N;
                                                WHEN "111" => StavToceni <= N;
                                                WHEN OTHERS=> StavToceni <= N;
                                        END CASE;
                                WHEN N   =>
                                        CASE AdresaProOtaceni IS
                                                WHEN "011" => StavToceni <= L1;
                                                WHEN "010" => StavToceni <= L1;
                                                WHEN "001" => StavToceni <= L1;
                                                WHEN "000" => StavToceni <= N;
                                                WHEN "101" => StavToceni <= R1;
                                                WHEN "110" => StavToceni <= R1;
                                                WHEN "111" => StavToceni <= R1;
                                                WHEN OTHERS=> StavToceni <= N;
                                        END CASE;

                                WHEN R1  =>
                                        CASE AdresaProOtaceni IS
```

```
                                        WHEN "011" => StavToceni <= N;
                                        WHEN "010" => StavToceni <= N;
                                        WHEN "001" => StavToceni <= N;
                                        WHEN "000" => StavToceni <= N;
                                        WHEN "101" => StavToceni <= R1;
                                        WHEN "110" => StavToceni <= R2;
                                        WHEN "111" => StavToceni <= R2;
                                        WHEN OTHERS=> StavToceni <= N;
                                END CASE;
                        WHEN R2  =>
                                CASE AdresaProOtaceni IS
                                        WHEN "011" => StavToceni <= R1;
                                        WHEN "010" => StavToceni <= R1;
                                        WHEN "001" => StavToceni <= R1;
                                        WHEN "000" => StavToceni <= R1;
                                        WHEN "101" => StavToceni <= R1;
                                        WHEN "110" => StavToceni <= R2;
                                        WHEN "111" => StavToceni <= R3;
                                        WHEN OTHERS=> StavToceni <= R1;
                                END CASE;
                        WHEN R3 =>
                                CASE AdresaProOtaceni IS
                                        WHEN "011" => StavToceni <= R2;
                                        WHEN "010" => StavToceni <= R2;
                                        WHEN "001" => StavToceni <= R2;
                                        WHEN "000" => StavToceni <= R2;
                                        WHEN "101" => StavToceni <= R2;
                                        WHEN "110" => StavToceni <= R2;
                                        WHEN "111" => StavToceni <= R3;
                                        WHEN OTHERS=> StavToceni <= R2;
                                END CASE;
                        END CASE;
                END IF;
END PROCESS ZmenyStavu;

-- ____________________________________________________________________________
--
--                          HLAVNI PROCES
-- ____________________________________________________________________________

HlavniProces: PROCESS (clk,reset)
variable  pomocna : integer range 0 to 7;

BEGIN
                IF (reset = '1') THEN
                        Krok1 <= 0;
                        Krok2 <= 0;
                        KrokBuben <= 0;
                        KrokOhrev <= 0;

                        Zobrazit   <= '0';
                        ZmenitStav <= '0';

                        StavVody <= SV1;
                        Spirala    <= SS1;


                ELSIF (clk'event and clk= '1') THEN
                        CASE StavToceni is
                                when L3 => pomocna:=1;
                                when L2 => pomocna:=3;
                                when L1 => pomocna:=7;
                                when N  => pomocna:=1;
                                when R1 => pomocna:=7;
                                when R2 => pomocna:=3;
                                when R3 => pomocna:=1;
                        END CASE;

                        Krok1 <= Krok1 + 1;
                        Krok2 <= Krok2 + 1;
                        KrokBuben <= KrokBuben + 1;
                        KrokOhrev <= KrokOhrev + 1;

                        IF Krok1 = 7 THEN                -- KROK1 udava, za kolik cyklu dojde ke zmene stavu rotace bubnu
                                Krok1 <= 0;
                                ZmenitStav <= NOT ZmenitStav ;
                                Zobrazit <= '1';
                        ELSE
                                Zobrazit <= '0';
                        END IF;


                        IF Krok2 = pomocna THEN          -- KROK2 udava, za kolik cyklu dojde k rotaci diod
                                Krok2 <= 0;
                                Zobrazit <= '1';
                        ELSE
                                Zobrazit <= '0';
                        END IF;
```

```
IF KrokBuben = 15 THEN              -- KrokBuben udava, za jak dlouho dojde ke zmene stavu vody
        KrokBuben <= 0;
        ---- VYSKA HLADINY V BUBNU ----
        CASE StavVody IS
                WHEN SV1 =>
                        CASE cerpadla IS
                                WHEN "01" => StavVody <= SV2;
                                WHEN "10" => StavVody <= SV1;
                                WHEN "11" => StavVody <= SV1;
                                WHEN "00" => StavVody <= SV1;
                        END CASE;
                WHEN SV2 =>
                        CASE cerpadla IS
                                WHEN "01" => StavVody <= SV3;
                                WHEN "10" => StavVody <= SV1;
                                WHEN "11" => StavVody <= SV2;
                                WHEN "00" => StavVody <= SV2;
                        END CASE;
                WHEN SV3 =>
                        CASE cerpadla IS
                                WHEN "01" => StavVody <= SV4;
                                WHEN "10" => StavVody <= SV2;
                                WHEN "11" => StavVody <= SV3;
                                WHEN "00" => StavVody <= SV3;
                                END CASE;
                WHEN SV4 =>
                        CASE cerpadla IS
                                WHEN "01" => StavVody <= SV5;
                                WHEN "10" => StavVody <= SV3;
                                WHEN "11" => StavVody <= SV4;
                                WHEN "00" => StavVody <= SV4;
                        END CASE;
                WHEN SV5 =>
                        CASE cerpadla IS
                                WHEN "01" => StavVody <= SV5;
                                WHEN "10" => StavVody <= SV4;
                                WHEN "11" => StavVody <= SV5;
                                WHEN "00" => StavVody <= SV5;
                        END CASE;
        END CASE;
END IF;


IF KrokOhrev = 23 THEN              -- KrokOhrev udava, za jak dlouho dojde ke zmene stavu teploty
        KrokOhrev <= 0;

        ---- OHREV SPIRALY ----
        CASE Spirala IS
                WHEN SS1 =>
                        CASE ProudSpiralou IS
                                WHEN "01" => Spirala <= SS1;
                                WHEN "11" => Spirala <= SS2;
                                WHEN "10" => Spirala <= SS1;
                                WHEN "00" => Spirala <= SS1;
                        END CASE;
                WHEN SS2 =>
                        CASE ProudSpiralou IS
                                WHEN "01" => Spirala <= SS2;
                                WHEN "11" => Spirala <= SS3;
                                WHEN "10" => Spirala <= SS1;
                                WHEN "00" => Spirala <= SS1;
                        END CASE;
                WHEN SS3 =>
                        CASE ProudSpiralou IS
                                WHEN "01" => Spirala <= SS3;
                                WHEN "11" => Spirala <= SS4;
                                WHEN "10" => Spirala <= SS2;
                                WHEN "00" => Spirala <= SS2;
                        END CASE;
                WHEN SS4 =>
                        CASE ProudSpiralou IS
                                WHEN "01" => Spirala <= SS4;
                                WHEN "11" => Spirala <= SS5;
                                WHEN "10" => Spirala <= SS3;
                                WHEN "00" => Spirala <= SS3;
                        END CASE;
                WHEN SS5 =>
                        CASE ProudSpiralou IS
                                WHEN "01" => Spirala <= SS5;
                                WHEN "11" => Spirala <= SS5;
                                WHEN "10" => Spirala <= SS4;
                                WHEN "00" => Spirala <= SS4;
                        END CASE;
        END CASE;
END IF;
```

```
                END IF;
END PROCESS HlavniProces;


-- ____________________________________________________________________________________
--
--                PROCES VYKONAVAJICI ROTACI DIOD (bubnu)
-- ____________________________________________________________________________________

RotaceBubnu: PROCESS (Zobrazit,reset)
BEGIN
        IF (reset = '1') THEN
                PomocnaToceni <= "0100";

        ELSIF (Zobrazit'event and Zobrazit= '1') THEN
                CASE StavToceni IS
                        WHEN L3 => PomocnaToceni <= PomocnaToceni rol 1; -- rotace
                        WHEN L2 => PomocnaToceni <= PomocnaToceni rol 1; -- doleva
                        WHEN L1 => PomocnaToceni <= PomocnaToceni rol 1;
                        WHEN N  => PomocnaToceni <= PomocnaToceni;
                        WHEN R1 => PomocnaToceni <= PomocnaToceni ror 1; -- rotace
                        WHEN R2 => PomocnaToceni <= PomocnaToceni ror 1; -- doprava
                        WHEN R3 => PomocnaToceni <= PomocnaToceni ror 1;
                END CASE;
        END IF;

        BubenDiody <= PomocnaToceni;    -- skutecne rozsviceni diod

END PROCESS RotaceBubnu;
-- ____________________________________ UKONCENI PROCESOVE CASTI ____________________________________

-- Zobrazeni diod indukujicich napousteni a vypousteni

        NapousteniVypousteni <= "11" WHEN cerpadla="11" and Reset='0' ELSE
                                "10" WHEN cerpadla="10" and Reset='0' ELSE
                                "01" WHEN cerpadla="01" and Reset='0' ELSE
                                "00" WHEN cerpadla="00" and Reset='0' ELSE
                                "00" ;
-- Indikace zavrenych dvirek

        DvirkaDioda  <= '0' WHEN Dvirka='0' and Reset='0' ELSE    -- sviti tehdy, jsou-li dvirka zavrena
                        '1' WHEN Dvirka='1' and Reset='0' ELSE
                        '0';
end behavioral;
```